# Power-Aware Multi-Objective Evolvable Hardware System on an FPGA

B. López, J. Valverde, E. de la Torre, T. Riesgo

## I. INTRODUCTION

EVOLVABLE hardware (EHW) implies hardware that can be reconfigured at run-time, so that its features of reconfiguration can be subjected to evolution, giving rise to a circuit or component that evolves towards an increasingly improved adaptation to certain conditions. When the evolution is achieved through a genetic algorithm, the implementation of the evolvable system involves: (1) defining which characteristics of the hardware act as genes – their combination would be the chromosome –, according to the bio-inspired approach to the design of a system; (2) introducing a procedure to yield an artificial evolution – which consists of the way the offspring are generated –, (3) and setting a mechanism of evaluation (a function to measure adaptation or *fitness*) that allows to range the individuals of the population according to their adaptation to the environment or fulfillment of certain objectives. Evolution is said to be intrinsic when it is performed by the system in charge of the execution itself, permitting online adaptation.

In order to build an intrinsic EHW system, it should be designed in a modular way so that some of its components can be modified and replaced by others during run-time. Online evolution on hardware enables a speed of search which could not be achieved by software evolution, and provides self-healing capabilities, as discussed in [1].

Modular and online modification of hardware is achieved by DPR, a feature that some manufacturers provide in order to access the configuration memory of an FPGA through the Internal Configuration Access Port (ICAP), allowing the device to reconfigure itself in an autonomous way. Among other alternatives of reconfiguration – like Virtual Reconfigurable Circuits (VRC) [2], where there exist hardware modules capable of performing different functions, which can be encoded by means of multiplexers –, DPR enables an efficient use of both the area and the resources of the FPGA. DPR implies the management of partial bitstreams, the files which contain the configuration data of a certain area within the FPGA.

An evolvable system may not head for a single objective, but for the optimization of a set of them. Most design environments deal with limitations, so that a trade-off between resources and success in a task is pursued, changing the point of view from effectiveness to efficiency. In this work, the design of a circuit able to know its own real power consumption in real time, and add the value of that parameter to the evolution process, is addressed. Evolution is performed according to a multi-objective scheme, which has been chosen to be based on Pareto optimality. The objective that stands for the effectiveness – and is intended to be optimized together with the power consumption – is the quality of a grayscale image filter based on a processing array, which is the particular application selected to illustrate the approach presented in this work.

The system is implemented on a low-power FPGA Spartan-6 LX by Xilinx, included in a high performance Wireless Sensor node called HiReCookie [3]. Although EHW is usually implemented on more powerful platforms like Virtex-5, restricted platforms with limited power budget may benefit from the adaptability of EHW, together with smaller power consumption. Some authors have also focused on the need of exploring low power consumption devices. The work presented in [4] tackles DPR on Spartan-6, introducing the features of ICAP and bitstream morphology in this platform, as well as the possibilities offered by its

granularity. In [5], the authors present their own tool for DPR on Spartan-6, applied to a novel "Fast start-up" method intended to configure the FPGA in two stages according to priority. A custom mechanism of self-reconfiguration on Spartan-6 is also addressed in [6], based on hard macros – to create a modular architecture – and access to a custom reconfiguration engine, compressed Parallel Configuration Access Port (cPCAP). In [7] the authors apply DPR on Spartan-6 to signal processing chains where the modules are sequentially loaded – in round-robin scheme – so that resources are optimally saved.

Multi-objective optimization on hardware has been explored in some works like the one presented in [8], where multi-objective evolution is applied to the design of combinational circuits, so that they fulfill a certain truth table and minimize the number of logic gates. The fitness function works in two successive stages: at the early one, only the validity of the circuit outputs is taken into account, while, once a feasible solution is found, the genetic algorithm tries to find the circuit with the lowest number of logic gates. Pareto optimality has been used in [9], where the authors apply multi-objective evolution to the design of digital filters – whose performance is usually multi-objective: constant amplitude in the pass-band, linear phase, etc. –, and in [10], where power distribution systems are designed according to a multi-objective evolutionary algorithm which determines how to distribute substations and connect the load nodes, minimizing both costs and non-supplied energy.

The optimization of power consumption through evolutionary algorithms has been addressed by some consulted works. However so far none of them performs a measurement of the actual power consumption in real-time; instead, estimation models are applied or offline-obtained values are used. In [11], the design of a circuit considering not only its effectiveness, but also complexity, power consumption and latency, is optimized; only solutions with 100% effectiveness are accepted. The optimization of an FFT processor is tackled in [12], according to the two objectives of Signal-to-Noise Ratio and power consumption; this last one is measured offline as part of the synthesis flow. An evolutionary algorithm is applied in [13] to optimize the order in which operations are executed in a microcontroller; the current consumed during each instruction of the microcontroller is measured offline so that values are available during evolution.

The paper is organized as follows. Section II provides a review of the preceding evolvable system – developed by the same research group – as well as the implementation platform. In Section III the creation of a reconfigurable structure on the Spartan-6 FPGA is addressed. The architecture of the system is described in Section IV. Section V provides an explanation of how the system has been designed to be power-aware. The multi-objective evolutionary algorithm is tackled in Section VI. Finally, Section VII shows the experimental results, and conclusions and future work are presented in Section VIII.

## II. BACKGROUND

### A. Single-objective evolvable system

The starting point of the system whose implementation is going to be described is a previous intrinsically evolvable system [14], implemented on a Virtex-5 FPGA and intended to optimize a single objective: the quality of a filtered grayscale image.

This system is based on a systolic array of processing elements (PEs), which perform different types of arithmetical operations and manage the data (pixels from the images) through biaxial flow. In order for these PEs to be interchangeable, they are provided with a common interface consisting of bus macros. The partial bitstreams of the different elements have to be extracted and stored in an external memory so that PEs can be loaded during reconfiguration.

The architecture of the system, as shown in Figure 1, consists of both static and reconfigurable parts. The static partition is comprised of the microprocessor (Microblaze), in charge of running the evolutionary algorithm; the memory blocks where images are stored; the reconfiguration engine

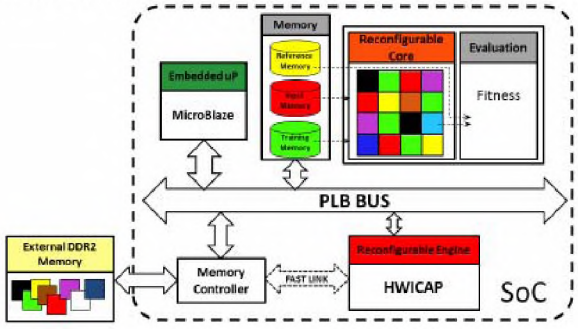


**Figure 1: Architecture of the single-objective evolvable system**

HWICAP, and a custom peripheral designed to manage the images, control the processing array, determine the fitness function, etc. The reconfigurable partition is the section occupied by the 4x4 size systolic array, whose inputs and output are selected in an evolutionary way, together with the functionality of the PEs.

As stated in the introduction, the evolutionary algorithm is characterized by three elements: genes, which in this system stand for the PEs, the selection of inputs and output and the filter's latency (number of cycles before fetching the output); evolution strategy, which is chosen as mutation-based generation of offspring from a single parent, and an evaluation mechanism, based on a fitness function which is here defined as the mean absolute error between the pixels of the original image and the filtered one. The evolutionary algorithm starts from an initial random parent and goes through a certain number of generations: in each of them, eight children are generated by mutation of the parent, and if any of them leads or equals the parent, it is chosen as the parent of the next generation.

### B. HiReCookie Platform

The platform on which the evolvable system has been implemented is a node of a Wireless Sensor Network called HiReCookie, developed in the same research centre. This node is intended for high-performance applications and responds to the necessity of higher processing capabilities in

Wireless Sensor Networks, where low power consumption has been traditionally the main concern. Thus, the architecture of these nodes has shifted from low profile microprocessors to more powerful platforms provided with specific hardware, which in HiReCookie consists of an FPGA, allowing for flexibility and DPR.

The node is characterized by a modular architecture, comprised of several layers: processing, communication, sensing and power supply, as shown in Figure 2.


**Figure 2: Modular architecture of the HiReCookie node**

The design of the node was oriented towards the possibility of implementing power management techniques [15]. Thus, it is divided into seven power islands – shown in Figure 3 – which can be selectively powered on or off by the microcontroller. In order to be able to measure the power consumption of each island, shunt resistors are placed in series with the power switches of the islands, so that their voltage drop can be measured and carried towards an Analog-to-Digital Converter (ADC). This feature has enabled the implementation of power-awareness in the multi-objective evolvable system.

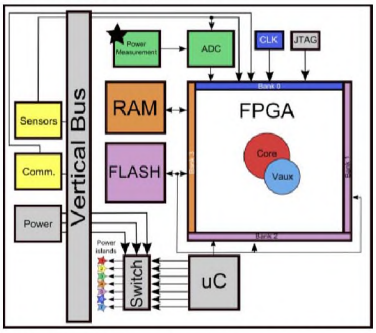


**Figure 3: Power islands of the HiReCookie node**

## III. RECONFIGURABLE STRUCTURE IN SPARTAN-6

An FPGA can be understood as the superposition of two virtual layers: one low-level layer where the logic is implemented, and one high-level layer, accessible for the user, comprised of the configuration memory, which stores the information about how the former layer is configured. The FPGA's capability of reconfiguring itself in an autonomous way derives from the existence of the ICAP, which provides internal access to the configuration memory. The transfer of data through ICAP is managed by the Xilinx IP Core HWICAP, and is limited by the maximum access frequency of ICAP, which in Spartan-6 is 20 MHz.

The low-level architecture of the FPGA has to be addressed so that its granularity can be assessed, allowing to design interchangeable modules and compatible interfaces. As well, correspondence between the FPGA fabric and the bitstream's structure can be determined. As already described in [16], the Spartan-6 FPGA on which this work is implemented is divided into 12 clock regions, each one comprised of 77 columns of different types (IOB, DSP, RAM…), among which CLB (Configurable Logic Block) columns predominate; CLBs are the object of reconfiguration and yield the FPGA's granularity. The minimum height of the module to be reconfigured corresponds to a clock region, since the minimum memory unit which can be addressed in the configuration memory is the frame, which describes part of a CLB column – each CLB column is described by a certain number of frames. Each frame consists of 65 words of 16 bits.

The processing elements are designed to occupy a height of one clock region and a width of 2 CLBs, which results in a processing array 4-clock-regions-tall and 8-CLBs-wide. To allocate the array within the FPGA fabric, two constraints have to be taken into consideration: the partition between static and reconfigurable areas, and the imposed distribution of certain cores along the FPGA's layout. As well, regularity has to be assessed, so that the array can be allocated on a homogeneous section. Due to layout's constraints, the only possible region for the array is one comprised of 8 CLB columns surrounded by RAM columns; as a result, being PEs 2-CLBs-wide, the bus macros between the elements are different depending on their location within the array. Thus, on extracting the bitstreams of the PEs, 3 bitstreams have to be extracted, corresponding to the three heterogeneous cases of the PEs being located on the left, middle or right part of the array. Figure 4 shows two captures of Xilinx FPGA Editor where the different implementation of the same element, depending on its location, is illustrated.

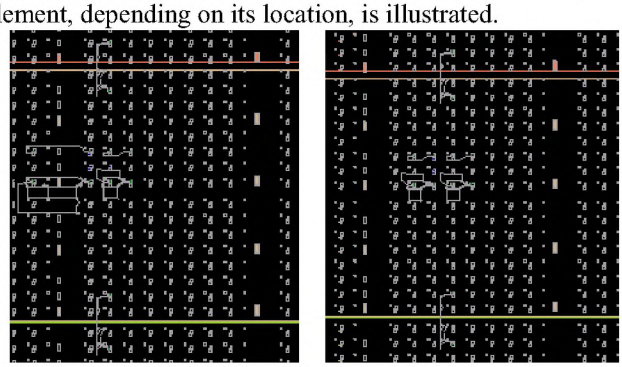

**Figure 4: FPGA Editor captures of a PE placed on the left and on the middle of a row within the array**

The processing elements are designed with Xilinx ISE, specifying constraints for the bus macros location and the area occupied by the PEs. The routing of the elements – it has to be confined to the reconfigurable module – and the automatic selection of the clock line are supervised – and, when opportune, modified – in FPGA Editor. Finally, bitstreams are extracted and stored in the Flash memory with a Xilinx SDK program.

## IV. ARCHITECTURE OF THE SYSTEM

An early version of the architecture was already tackled when implementing the single-objective evolvable system on Virtex-5, and most of it has been migrated to the new platform, incorporating some additional features.

The static region is the part of the system in charge of providing data and fetching the results from the systolic array. The design of the hardware is performed with Xilinx

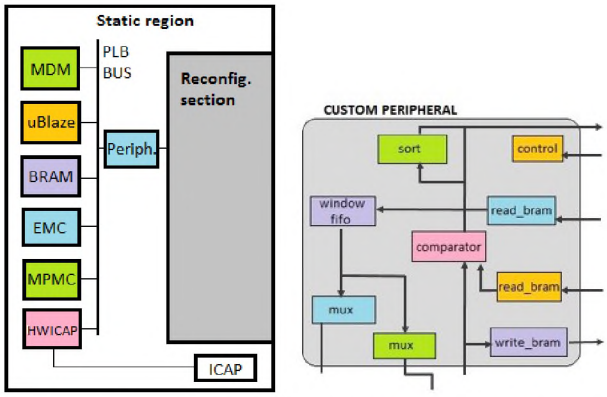


**Figure 5: Architecture of the system and the custom peripheral**

XPS and consists in the implementation of the following blocks: RAM internal memory blocks, memory controllers for RAM and Flash, HWICAP and the custom peripheral introduced in Section II. This peripheral is comprised of several modules which materialize the filtering process in the convenient sequential way; the covered functions include: reading and writing of the internal RAM memories through buffers, generation of a 3x3-pixel-window which goes through the image, configuration of the input and output multiplexers, computation of the fitness function and registration of the outputs corresponding to different latencies of the processing array. As well, a control module is included in order to referee the filtering process, triggering the other modules so that their latencies are taken into consideration. The architecture of the system is schematized in Figure 5.

**Figure 6: View of the system from FPGA Editor**

In order to incorporate the real power consumption as a parameter during evolution, creating a power-aware system, additional modules have to be implemented in the custom peripheral. These blocks will be addressed in detail in Section V.

One of the most relevant aspects of the system implementation is the strategic layout within the FPGA. The location of the modules is constrained through both Xilinx ISE and Plan Ahead, and takes into account the connectivity needs of the blocks, as well as the boundaries imposed by the reconfigurable region. Xilinx FPGA Editor allows to supervise the design, which has to fulfill certain conditions to be reconfigurable. The final view of the system in FPGA Editor turns out as shown in Figure 6, where the highlighted empty area corresponds to the systolic array.

## V. POWER-AWARENESS

The evolutionary approach to a problem lies in disregarding any model of the system; instead, based on some basic specifications, the search space is explored so that the best solution is found through randomness and the imitation of natural selection. Subsequently, an evolutionary approach to power optimization should not be based on any model that splits the power consumption into different formalized contributions.

An approach that consists in defining such model has been found in [17], where power consumption is divided into three components: static power consumption due to the leakage current (caused by the parasitic diodes in CMOS gates) and static current which depends on the input voltage; dynamic power consumption caused by the charge and discharge of the effective load capacitance of each node, and dynamic power consumption caused by the short-circuit current during the switching transient. In addition, a power model is developed to derive the power consumption from data of the implementation resources of the FPGA.

Power estimation models have been developed by some research groups, like [18], based on the Place&Route tool VPR, and some are provided by Xilinx, which offers two different tools: Xilinx Power Estimator (XPE) and Xilinx Power Analyzer (XPower). XPE is based on a spreadsheet which stores information about the utilized resources after the mapping process; it yields fast but imprecise results. XPower is more accurate, as it takes into consideration the physical layout and the simulated toggle rates.

In addition to hardware acceleration, online evolution presents another motivation: if the evolution was performed offline, although results of quality of filtering could be determined, results for power consumption could only be estimated according to some model. Intrinsic evolution makes it possible to know the real power consumption, given that the device can be power-aware.

As well, silicon process variation could be capable of producing sufficiently different power consumption between one device and another. The proposed methodology would permit case-by-case power optimization.

### *A. Experimental analysis*

The first step to approach the power consumption measurement consisted in observing the power consumption profile of the FPGA during the evolutionary process by means of an oscilloscope. What was pursued was to know the shape of the power consumption readings that would be performed in the end by the embedded ADC, so as to choose the most suitable measurement method.

The power island from which the power consumption reading is obtained is the FPGA core (which includes the Microblaze) – it will be further described in the next section. The power consumption profile was acquired for two chromosomes – two configurations – that were loaded on the FPGA, triggering the filtering process afterwards. Power consumption peaks were observed during both

reconfiguration and filtering, as shown in Figure 7. The power consumption to be optimized is the one corresponding to the filtering stage, as a system working in normal operation mode (in contrast to evolvable mode) will be filtering images continuously.

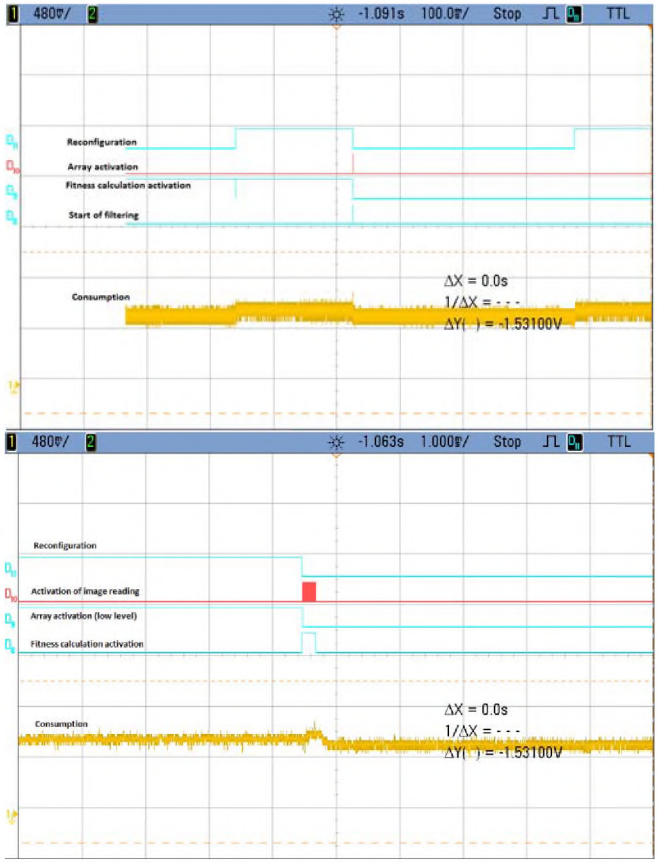


**Figure 7: Oscilloscope captures of power consumption during the different stages of the evolutionary process**

### *B. Implementation of power-awareness*

The HiReCookie node is provided with an ADC that allows both to acquire the sensors readings and to measure the power consumption of every power island. The addition of modules to the custom peripheral in order to control the operation of the ADC and trigger the power consumption measurements allows to incorporate the dynamic power consumption during filtering to the evolutionary algorithm as a second parameter, together with fitness.

#### *ADC interface and power measurement circuitry*

The model of the ADC is AD7928 from Analog Devices and is provided with eight analog inputs and 12-bit resolution. As shown in Figure 8, some of the inputs are destined to convert the power consumption from the power islands. Communication between the FPGA and the ADC is managed through SPI protocol. Xilinx provides an IP Core to control devices based on SPI interface; however, obtaining the power consumption measurements via software was discarded, as it is necessary to parallelize the measurements and the operation of the system and delimit very precisely the measurement time, synchronized with other hardware modules. Thus, the interface is implemented on hardware, in the form of additional modules in the custom peripheral.

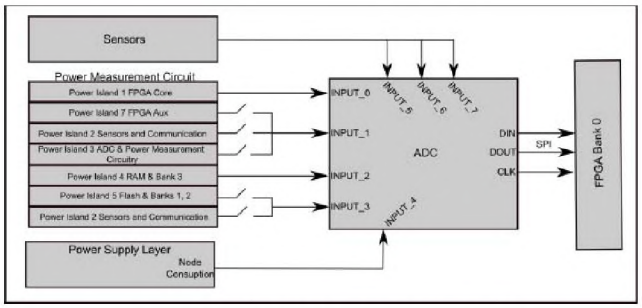


**Figure 8: Connections of the ADC**

To obtain the value of power consumption, the current demanded by the FPGA core is led to a shunt resistor, so that the voltage drop, proportional to the power, is amplified by an instrumentation amplifier and then carried to the ADC. When dealing with consumption, energy is the relevant magnitude to pay attention to, rather than power; however, as the filtering time is constant and not dependent on the chromosome, measuring the power is equally representative.

#### *Control of the ADC from peripheral*

During each evaluation of a chromosome, the ADC is asked by the Microblaze to perform two measurements: the initial power consumption, corresponding to the static power consumption before the reconfiguration of the systolic array, and the power consumption during filtering. Power measurements are chosen to be differential (difference between the initial power consumption and its value during filtering) in order to decouple the measurements from the thermal state of the FPGA (if the temperature rises, the voltage drop through the shunt resistor increases, yielding a higher value of power consumption during filtering).

Another consideration to be taken into account is the noise in the measurements, as the ADC is exposed to electromagnetic interferences. Implementing some filter helps to obtain more significant values. A moving-average filter enables to clarify the tendency of a signal (low-pass effect).

The modules added to the custom peripheral comprise a controller of the ADC, a control module and a processing block.

The ADC controller generates the SPI protocol and configuration signals and delimits the data transfer cycles, ensuring that the timing constraints are fulfilled. The processing block receives several enable signals, depending on which it performs different operations upon the power consumption readings, such as moving-average filtering and averaging a certain number of measurements.

The control module is a finite-state machine which commands to perform a certain number of measurements during a specific time interval, depending on which measurement is made: initial power consumption or power consumption during filtering.

## VI. MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM

In multi-objective problems, there tends to be a conflict among the objectives, so that an outstanding performance in one of them is usually detrimental to other goals. There are

two possible approaches to a multi-objective problem: address the problem as an extension of the mono-objective algorithm, combining the individual objective functions into a single compound function (by means of the definition of weights), or apply the Pareto optimality, according to which a solution is optimal (also called "non-dominated solution") if no other solution leads it in one objective without being detrimental to other objectives. The first approach is discarded due to its poor robustness, since slight variations in the values of the weights can lead to very different solutions [19]. The Pareto-based algorithms pursue to find the Pareto-optimal set of solutions; in fact, a representative subset, due to the size of the complete set.

Among the existing multi-objective evolutionary algorithms, NSGA II [20] has been selected. This algorithm is characterized by the following strong points [21]: elitism, moderate computational complexity and the lack of the need to specify a "sharing-parameter" – as needed in the fitness-sharing method. The evolutionary process is guided by two features: non-dominance rank and distance. Non-dominance relates to the fulfillment of Pareto optimality, whereas distance is a measure of how far a solution is from the other ones. Non-dominated solutions are prioritized and, under the same conditions, solutions located in less densely populated regions would rather be selected. In this way, diversity is preserved. Selection of type (μ+λ) is employed.

### A. *Non-dominated sorting*

The evolutionary process starts from an initial random population of μ parents, where the first Pareto front (the set of non-dominated solutions) has to be obtained; for that purpose, each solution has to be compared with the other ones in terms of both fitness and power consumption. Afterwards the successive Pareto fronts have to be identified, following the same procedure, which is known as non-dominated sorting.

Two entities have to be calculated for each solution p:

1) $n_p$, the dominance count, defined as the number of solutions dominated by the solution p
2) $S_p$, the set of solutions that the solution p dominates

The solutions whose dominance count equals zero ($n_p$=0) belong to the first Pareto front. For each solution p with $n_p$=0, each member q of its set $S_p$ is visited, reducing its dominance count by one. After this process, if any solution turns out with dominance count zero, it is integrated in the second Pareto front. Repeating the process allows to identify the different Pareto fronts, until a void front is encountered. At the end of the sorting, the non-dominance rank of every solution is known – solutions from the first Pareto front have rank 1, solutions from the second front have rank 2, and so on and so forth.

### B. *Diversity Preservation*

As a measure of the density of solutions surrounding a particular solution, the parameter "distance" of the solution is calculated as the sum of the differences in each of the objectives between the two neighbouring solutions (being the solutions ranked in a graph that represents both objectives, as shown in Figure 9).

Solutions are firstly ranked according to fitness. If they are placed at the ends, they are assigned infinite distance.

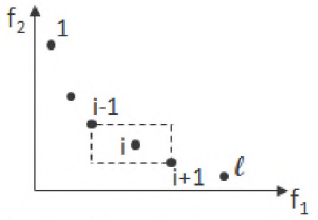


**Figure 9: Ordering of solutions to calculate distances**

Otherwise, the distance is calculated as the difference in fitness (absolute value) between the neighbouring solutions, divided between the maximum fitness gap within that set of solutions (so that a normalized distance between 0 and 1 is obtained). The process is repeated by ranking the solutions according to the power consumption and adding the newly calculated distances (based on power consumption) to the formerly obtained ones.

### C. *Application of NSGA II to the power-aware system*

Initially, a random population of four parents is generated. For each of them, the systolic array is updated with that given configuration; static power consumption is measured, the filtering process is triggered and finally the values of fitness and power consumption during filtering are obtained. Based on those two objectives, Pareto fronts within this population are identified and distances are calculated. Afterwards the offspring (as well a population of size 4) are generated, by binary tournament between two randomly picked up parents and mutation applied to the winning parent. Then the global population, of size 8, is formed gathering parents and children, and again Pareto fronts and distances are obtained within this population. Finally, the future parent population is selected in order of Pareto front, taking into account the diversity criterion in the event of draw. A single generation's execution is schematized in Figure 10. The Pareto-based multi-objective evolutionary algorithm does not penalize the results of a single objective, as it does not constrain the exploration of the search space. Convergence to low fitness values is preserved, relative to the results obtained in the single-objective system.

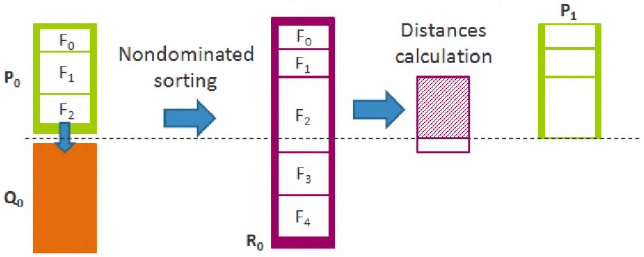


**Figure 10: Execution of a single generation within the multi-objective evolutionary algorithm**

## VII. EXPERIMENTAL RESULTS

At a first stage the evolutionary process was launched to filter a grayscale image with a salt-and-pepper-noise level of 5%, employing a mutation rate of 3 (K=3), which means, introducing three changes on every mutation of a chromosome.

The evolution comprised 100,000 generations and was executed 20 times. Given the timing information of Table I, it takes approximately 40 minutes to perform an evolutionary process of 100,000 generations; nevertheless, a

much lower number of generations is usually enough to reach sufficiently good and conclusive results. As well, a significant acceleration is achievable for other FPGA families whose ICAP ports can reach much higher speed, with comparable timings for both evaluation and reconfiguration (in the order of hundreds of microseconds).

| Process | Time |
|---|---|
| Reconfiguration of one PE | 8 ms |
| Filtering process | 163 µs |

**Table I: Time employed by the Spartan6-based platform to perform different stages of the evolutionary process**

One of those executions is shown in Figure 11, where the final population of the evolutionary process, of size 4, is represented by 4 points in a graph whose axis are fitness and power consumption (differential value over static power consumption). The resulting filtered images that correspond to each point are also shown.

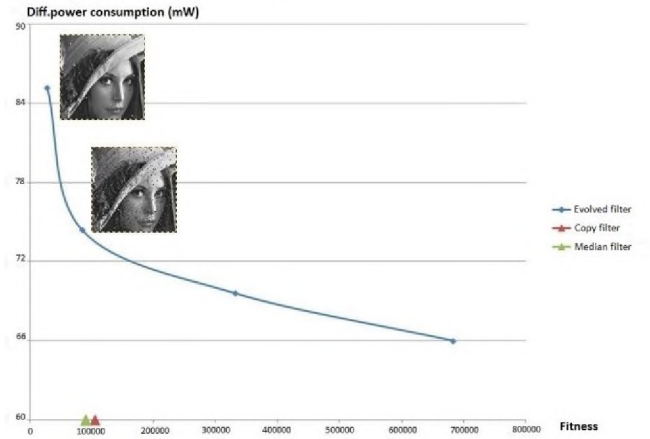


**Figure 11: Pareto front yielded by an evolutionary process applied to an image with 5% noise and K=3**

It can be observed how the shape of the set of solutions resembles a typical Pareto front. Two thresholds are represented in the figure, corresponding to: the median filter, usual reference to assess the quality of filtering, and the copy filter, which passes the input to the output without modifying the image. Solutions beyond the copy filter's threshold are rejected since they do not perform any filtering function. In this single execution, two feasible solutions are obtained, with a significant difference both in fitness and power consumption. Thus, in a very low power application the best fitness solution could be discarded.

The difference in power consumption was checked by designing a non-evolvable system configured with a filter operating in normal mode (filtering images continuously), and the battery level was monitored during its operation. The results are shown in Figure 12, where it can be perceived how the battery voltage falls more quickly in the configuration of the best fitness filter (red coloured in the figure).

Results of the set of experiments show that the difference in the battery lifetime among configurations ranges from 5% to 10% approximately. These relatively low values were expected, as only the functionality of the PEs is different from one configuration to another, while the system holds an equivalent and fixed-size structure.

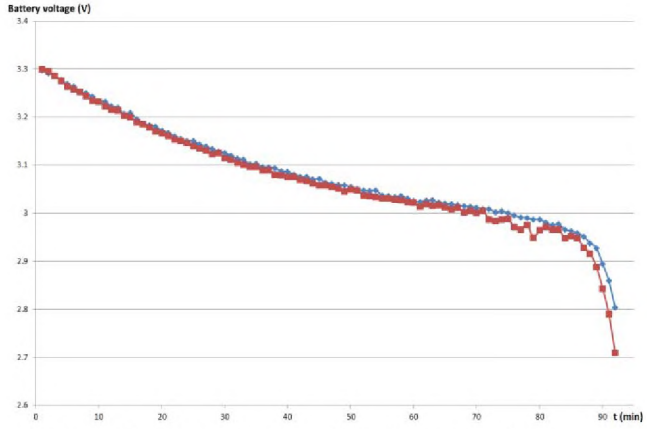


**Figure 12: Measurement of the battery voltage as a function of time for two different configurations**

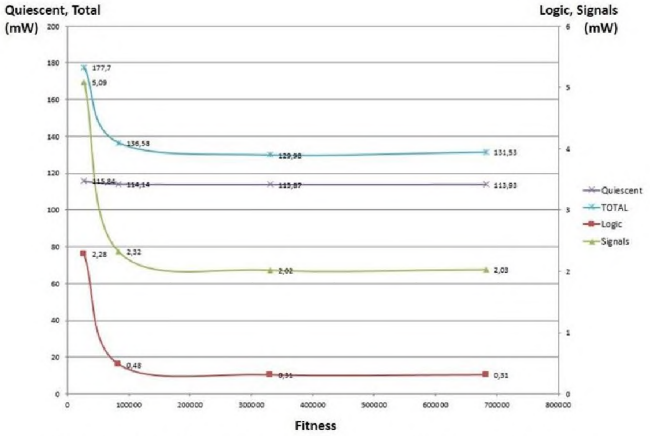


**Figure 13: Power estimations of XPower for the configurations obtained in the Pareto front**

It was also tested if these results could be predicted from the physical implementation of each filter. For that purpose, each of the systems corresponding to the solutions obtained in the Pareto front was simulated with XPower. As it can be seen in Figure 13, the estimated points draw again the shape of a Pareto front. Power is split into different contributions that comprise quiescent power and power associated with logic occupation and signals activity.

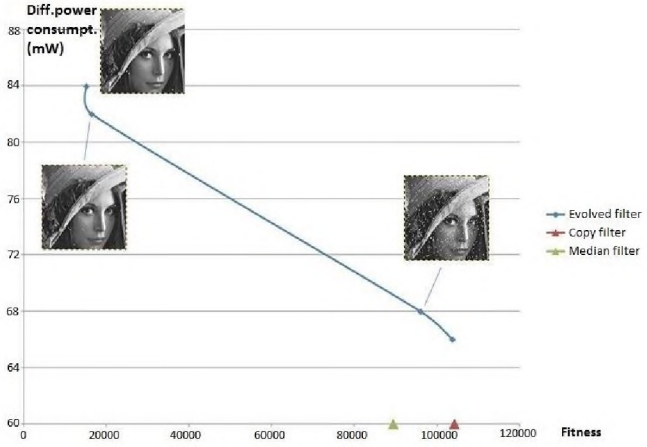


**Figure 14: Pareto front yielded by an evolutionary process applied to an image with 5% noise and K=4**

XPower results can help to corroborate the validity of our methodology, as the tendency shown by the experimental Pareto front demonstrates correlation to a physical model. However, XPower would not be able to predict the exact real power consumption, due to its inherent lack of accuracy (it is based on simulated activity rates). As well, the system which is simulated is not the original one, as it does not operate on evolvable mode, but filtering images with a fixed configuration.

Additional experiments were made changing the noise level and the mutation rate. Some of the executions are represented in Figure 14 and Figure 15.


**Figure 15: Pareto front yielded by an evolutionary process applied to an image with 10% noise and K=4**

## VIII. CONCLUSIONS AND FUTURE WORK

An evolvable system has been implemented on a low-power FPGA included in a Wireless Sensor node. A multi-objective evolutionary algorithm has been successfully employed to find optimal solutions in terms of both functionality and power consumption. As a demonstration, it has been applied to the filtering of noisy grayscale images. In contrast to other methods, power consumption is not estimated or measured offline, but intrinsically measured in real time. This allows to add the real power consumption – associated with a particular configuration – during filtering as a parameter in the evolution. Pareto optimality enables to come up with a set of optimal solutions, among which the designer can choose according to the specific priorities.

As for future work, multi-objective evolution is intended to be applied to a scalable evolvable system, so that the growth of the processing tissue during evolution is guided by Pareto optimality. Power consumption differences are expected to be higher than in the fixed-size system, where only the functionality of the PEs is different from one configuration to another.

## ACKNOWLEDGEMENT

This work was supported by the Spanish Ministry of Economy and Competitiveness under the project DREAMS (Dynamically Reconfigurable Embedded Platforms for Networked Context-Aware Multimedia Systems) with number TE2011-28666-C04-02.